"NONPARAMETRIC METHODS ARE TOO SLOW?" (SOME THOUGHTS ON MODEL SPECIFICATION)

JEFFREY S. RACINE

Statistical Framework

Consider a stochastic relationship involving a set of predictors $X \in \mathbb{R}^k$ and a response $Y \in \mathbb{R}^1$. Suppose interest lies in the average value of Y given $X = x$ which we denote $g(x)$ for a location-scale model given by $Y = g(X) + \sigma(X)\epsilon$.

It is not uncommon to encounter practitioners wielding classical regression methods who presume, implicitly or explicitly, that

(1) the underlying data generating process is linear in variables and additively separable, or
(2) the model they have written down is properly specified, or
(3) the model others have used must be properly specified and therefore requires no additional checking (though they may be using a different data frame).

Each of these scenarios share a common thread - the non-trivial process of model selection has been completely sidestepped and ignored (why?). Naturally, if the model one has in hand is the only model you are going to entertain, then all science is done (as long as you or others don't look too closely under the hood). You have in effect peered into the cosmos and divined the true nature of the relationship at work, expressed it as a concise mathematical model, and fit it using classical techniques. And since you are using simple least squares or perhaps maximum likelihood, estimation is typically trivial and computation of the model is often virtually immediate. So it follows naturally that this would become your benchmark for expected computational complexity, and when you confront alternative approaches that require relatively more computation time, you let loose the oft heard refrain 'these nonparametric methods are so slow!'.

Date: July 8, 2014.

A Simple Linear-in-Variables Parametric Model. Practitioners often gravitate towards estimating unknown relationships using the simple linear additive model that we consider below (we include timing information for comparison purposes). Suppose we consider $n = 1000$ observations on (Y, X_1, X_2, X_3), all of which are continuous random variables. We fit a simple linear additive model of the form

```
> t.lm <- system.time(model.lm <- lm(y~x1+x2+x3,data=mydat))
> t.lm
   user  system elapsed
  0.007   0.001   0.010
```

Whether this simple model is consistent with the data being analyzed, however, is another matter altogether. Were one in the least bit concerned with potential model misspecification, naturally one would conduct a test for correct specification, so let's do this using Ramsey's (1969) test.

```
> ## Use Ramsey's RESET test from the lmtest package
> resettest(model.lm)

        RESET test

data:  model.lm
RESET = 6.9971, df1 = 2, df2 = 994, p-value = 0.0009602
```

Oops. This model is rejected by the data (P-value of 0.0009602338). At this point I direct the reader to my rant 'Find Your Keys Yet?', but point out that it is not uncommon to encounter academics presenting work based on models they implicitly presume are correctly specified but for which absolutely no specification testing has been undertaken whatsoever (after all, when you have divined the laws of nature, why waste time on such mundane things?).

A Linear Parametric Model Selected via Schwarz-Bayes Information Criterion. So next suppose one admits model uncertainty but wishes to remain in a parametric framework. We could conduct stepwise model selection guided by the Schwarz-Bayes Information Criterion (SBIC, Schwarz (1978)). Here you have to provide an upper bound on the model complexity along with a lower bound. By way of illustration suppose we consider models spanning the simple linear model given above but also consider all models involving polynomials up to a given order along with interactions up to a given order. So here we are treading on the well-worn path of SBIC parametric model selection.

```
> ## Conduct Schwarz-Bayes parametric model selection
> library(MASS)
> ## Specify the maximum complexity for the parametric model (i.e.
> ## polynomial of order 10 in each variable with interactions up
> ## to order 3)
> formula.upper <- formula(y~(x1+x2+x3
+                           +I(x1^2)+I(x2^2)+I(x3^2)
+                           +I(x1^3)+I(x2^3)+I(x3^3)
+                           +I(x1^4)+I(x2^4)+I(x3^4)
+                           +I(x1^5)+I(x2^5)+I(x3^5)
+                           +I(x1^6)+I(x2^6)+I(x3^6)
+                           +I(x1^7)+I(x2^7)+I(x3^7)
+                           +I(x1^8)+I(x2^8)+I(x3^8)
+                           +I(x1^9)+I(x2^9)+I(x3^9)
+                           +I(x1^10)+I(x2^10)+I(x3^10))^3)
> ## Perform Schwarz-Bayes model selection, and compute the estimation time
> t.BIC <- system.time(model.BIC <- stepAIC(lm(y~.,data=mydat),
+                                           scope=list(upper=formula.upper,
+                                                      lower=~1),
+                                           k=log(length(mydat$y)),
+                                           trace=0))
> t.BIC
   user  system elapsed
156.670   2.930 159.597
```

Local Linear Kernel Regression with Cross-Validated Bandwidths. Next we consider local linear kernel regression. We refrain from presuming we know the bandwidths (or using ad hoc values) and let cross-validation determine the appropriate amount of smoothing (i.e. the appropriate non-parametric model).

```
> ## Now conduct local linear kernel regression
> ## using the np package
> t.np <- system.time(model.np <- npreg(y~x1+x2+x3,
+                                       nmulti=1,
+                                       regtype="ll",
+                                       data=mydat))
> t.np
```

```
   user  system elapsed
 11.608   0.010  11.630
```

Regression Splines with Cross-Validated Degree and Knots. Last, we consider regression B-splines. We also refrain from presuming we know the optimal spline degree or knots and again let cross-validation determine the appropriate amount of smoothing (i.e. the appropriate nonparametric model).

```
> ## Now conduct a regression spline using the crs
> ## package
> t.crs <- system.time(model.crs <- crs(y~x1+x2+x3,
+                                       nmulti=1,
+                                       data=mydat))
> t.crs
   user  system elapsed
 47.009   1.960  48.985
```

As you can see, the estimation times for each approach (parametric with Schwarz-Bayes model selection, nonparametric kernel regression with cross-validated model selection, and nonparametric regression spline with cross-validated model selection) are 156.67, 11.608 and 47.009 seconds, respectively. So now the tables are turned, which to some may be surprising.

Reflections on Computation Time

To my way of thinking, the proper computational comparison is between model selection in a parametric framework versus model selection in a nonparametric framework and not, as is often done, between one model you have pulled out of thin air and assert to be the divine truth versus nonparametric methods that deliver consistent estimates in a sound statistical framework where virtually all computation is for the purpose of model specification. The stance that the presumed parametric model is 'true' and therefore we can immediately jump to interpretation without subjecting the model to any specification testing whatsoever is simply not defensible. Yet this is not uncommon. Model selection is a fundamental aspect of sound data analysis that cannot be ignored.

After all, using cross-validation to obtain appropriate bandwidths for kernel regression is nonparametric model selection. And using cross-validation to determine the number of knots or the spline degree for regression splines

is also model selection. Of course, there is a rich history of using cross-validation for parametric model selection as well. If researchers took the process of model specification seriously rather than pretending they miraculously pulled a model from the space of all possible models (which is dense) and have managed to select and write down the correct one, they would appreciate that this fundamental and crucial step in fact is non-trivial.

By its very nature model selection is computationally burdensome. And as was demonstrated above, nonparametric models can in fact be *less* of a computational burden than many would ever realize *if* their reference point was parametric model specification. This is because they have never attempted to do so in a parametric setting to begin with, having settled for the false security that comes with believing that any particular parametric model they write down happens to be the 'true' model. To paraphrase Aman Ullah, finding the correct model, like achieving nirvana, is known not to be an easy task.

In fact, I think a more relevant question to pose to someone asking 'why are nonparametric methods so slow?' is the question 'why do you suppose it is that your approach seems to be so fast?' Is it because you totally sidestepped step one (i.e. the model specification exercise) and therefore have the (wrong) impression that it is your method that is fast, when in fact the opposite might be the case? For that matter, give me the 'optimal' bandwidths or the 'optimal' number of knots and spline degree and nonparametric methods can be almost as fast as computing the 'true' model (after all, regression splines and kernel methods are simply weighted least squares estimators). Combine this with the fact that the low order raw polynomials often found in applied research can only approximate a small subset of models that the nonparametric methods can, and the fact that in order for parametric model selection to be consistent the 'true model' must be nested in the set of approximating models, and it does give one cause to reflect on the state of much applied work in our field.

Let me finish with the disclaimer that in no way am I asserting superiority of nonparametric over parametric approaches. I view all models as approximations and as such model uncertainty plays a central role in my world view. Practitioners ought to have a variety of tools at their disposal, each one tailored to particular tasks, and parametric approaches are invaluable. Furthermore, whether or not a nonparametric model will perform any better

than a misspecified parametric model is a very good question indeed, and there are no pat answers to this important issue.

Find your keys yet?

Appendix A. Linear Parametric Model Summary

```
> summary(model.lm)
Call:
lm(formula = y ~ x1 + x2 + x3, data = mydat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.99919 -0.75547 -0.02041  0.75783  2.59837

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.54702    0.09955   5.495 4.97e-08 ***
x1           0.42467    0.11085   3.831 0.000136 ***
x2          -1.33839    0.10901 -12.278  < 2e-16 ***
x3           0.99552    0.11159   8.921  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.02 on 996 degrees of freedom
Multiple R-squared:  0.2011,        Adjusted R-squared:  0.1986
F-statistic: 83.55 on 3 and 996 DF,  p-value: < 2.2e-16
```

Appendix B. Schwarz-Bayes Parametric Model Summary

```
> summary(model.BIC)

Call:
lm(formula = y ~ x1 + x2 + I(x1^2) + I(x2^10) + I(x2^2) + I(x2^4) +
    I(x3^3) + I(x1^6) + x1:x2 + x1:I(x1^2), data = mydat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.43081 -0.32978 -0.00733  0.31810  1.82629

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.86209    0.10706   8.053 2.32e-15 ***
x1           -1.09933    0.88156  -1.247    0.213
x2           11.67298    0.58869  19.829  < 2e-16 ***
I(x1^2)     -21.51454    2.89872  -7.422 2.48e-13 ***
I(x2^10)     -2.38599    0.42588  -5.603 2.74e-08 ***
I(x2^2)     -26.34970    1.19993 -21.959  < 2e-16 ***
I(x2^4)      17.35595    1.03439  16.779  < 2e-16 ***
I(x3^3)       1.06251    0.05634  18.860  < 2e-16 ***
I(x1^6)      -9.74193    0.83032 -11.733  < 2e-16 ***
x1:x2         0.96766    0.18831   5.139 3.33e-07 ***
x1:I(x1^2)   32.11512    2.84362  11.294  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5033 on 989 degrees of freedom
Multiple R-squared:  0.8069,        Adjusted R-squared:  0.805
F-statistic: 413.3 on 10 and 989 DF,  p-value: < 2.2e-16
```

Appendix C. Nonparametric Kernel Regression Model Summary

```
> summary(model.np)
Regression Data: 1000 training points, in 3 variable(s)
                      x1         x2        x3
Bandwidth(s): 0.07267996 0.06776437 0.2650485

Kernel Regression Estimator: Local-Linear
Bandwidth Type: Fixed
Residual standard error: 0.4642253
R-squared: 0.8381235

Continuous Kernel Type: Second-Order Gaussian
No. Continuous Explanatory Vars.: 3
```

Appendix D. Nonparametric Regression Spline Model Summary

```
> summary(model.crs)

Call:
crs.formula(formula = y ~ x1 + x2 + x3, data = mydat, nmulti = 1)

Indicator Bases/B-spline Bases Regression Spline

There are 3 continuous predictors
Spline degree/number of segments for x1: 7/8
Spline degree/number of segments for x2: 6/4
Spline degree/number of segments for x3: 2/1
Model complexity proxy: degree-knots
Knot type: quantiles
Basis type: additive
Pruning of final model: FALSE
Training observations: 1000
Rank of model frame: 26
Trace of smoother matrix: 26

Residual standard error: 0.5047 on 974 degrees of freedom
Multiple R-squared: 0.8088,   Adjusted R-squared: 0.8039
F-statistic: 164.8 on 25 and 974 DF, p-value: 0

Cross-validation score: 0.26187075
Number of multistarts: 1
Estimation time: 46.9 seconds
```

References

Ramsey, J. (1969), 'Tests for specification error in classical linear least squares regression analysis', *Journal of the Royal Statistical Society, Series B* **31**, 350–371.

Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* **6**, 461–464.